A NOTE ON FINDING ALL SHORTEST PATHS

by

Fred Glover*
D. Klingman
A. Napier

CENTER FOR CYBERNETIC STUDIES
The University of Texas
Austin, Texas 78712

Research Report
CS 88

A NOTE ON FINDING ALL SHORTEST PATHS

by

Fred Glover*
D. Klingman
A. Napier

May 1972

* University of Colorado

This research was partly supported by a grant from the Farah Foundation and by ONR Contracts N00014-67-A-0126-0008 and N00014-67-A-0126-0009 with the Center for Cybernetic Studies, The University of Texas. Reproduction in whole or in part is permitted for any purpose of the United States Government.

CENTER FOR CYBERNETIC STUDIES

A. Charnes, Director
Business-Economics Building, 512
The University of Texas
Austin, Texas 78712

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)

Center for Cybernetic Studies
The University of Texas

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

3. REPORT TITLE

A Note on Finding All Shortest Paths

4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Center for Cybernetic Studies

5. AUTHOR(S) (First name, middle initial, last name)

F. Glover A. Napier
D. Klingman

6. REPORT DATE	7a. TOTAL NO OF PAGES	7b. NO OF REFS
May 1972	14	10

8a. CONTRACT OR GRANT NO	9a. ORIGINATOR'S REPORT NUMBER(S)
NR-047-021	Center for Cybernetic Studies Research Report 88
b. PROJECT NO	
c. N00014-67-A-0126-0008	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d. N00014-67-A-0126-0009	

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Office of Naval Research (Code 434) Washington, D. C.

13. ABSTRACT

We present a procedure for finding the shortest route between all pairs of nodes in a network. Our approach is quite simple, but is highly efficient for a special class of network problems. In particular, this class of networks subsumes a number of networks which appear in real world application. For instance, the Polish government has developed a network model for water resource analysis which is in this class of networks.

14

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0102-014-6700

Unclassified

Security Classification

D-23822

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Shortest Path Problems						
Network Problems						
Transportation Problems						
All Shortest Path Problems						

Abstract

We present a procedure for finding the shortest route between all pairs of nodes in a network. Our approach is quite simple, but is highly efficient for a special class of network problems. In particular, this class of networks subsumes a number of networks which appear in real world application. For instance, the Polish government has developed a network model for water resource analysis which is in this class of networks.

1 Introduction

One of the primary types of shortest route problems is the determination of the shortest route between all pairs of nodes in a network. Although others (e.g., Floyd [4], Dantzig [1], Hu and Torres [8], Land and Stairs [9], and Yen [10]) have proposed very efficient procedures for solving such problems we present a different approach.[1] Our approach is quite simply, but is highly efficient for a special class of networks. In particular, this class of networks seems to subsume a number of networks which appear in real world applications. For instance, the Polish government has developed a network model for water resource analysis which is in this class of networks. Also, the Texas Water Development Board has a network model for water resource simulations which is in this class.[2] Still other members of this class include product distribution problems for large industries such as the automobile industry.

2 Background and Mathematical Development

Hu and Torres [8] developed a decomposition algorithm, which is a modification of Hu's algorithm [6], for obtaining the shortest path between all pairs of nodes in a sparse network. Yen [10] has modified their procedure to enhance its computational efficiency. The algorithms developed by Hu and Torres, and Yen assume a network N that contains no negative cycles and that has been decomposed into m linear overlapping node sets $\overline{A}$, $\overline{B}$, ..., $\overline{H}$ or $N = A \cup x_A \cup B \cup x_B \cup \ldots G \cup x_G \cup H$ where x_A, x_B, ..., are the minimum cut sets for A, $A \cup x_A \cup B$, ..., respectively. (See Figure 1.) Furthermore, as indicated by Figure 1, a path of finite distance can exist from a node in A to a node in H via nodes in x_A, B, ..., x_G. Thus, these models are designed to reflect the situation in which it is likely that a path exists from any node to any other node in the network, and the methods of [6, 8, 10] expend

computational effort to exploit this fact. It is our experience, however, that such paths will not exist for a large number of node pairs and further that such paths only exist for node pairs in the same "region" of the network.

Consequently, it seems more appropriate, at least in certain applications, to consider a decomposition of the network that arises by identifying node sets such that every node in a given set lies on a directed path from one or more "origin" nodes in this set. For a collection of node sets so determined, we may associate with each one a unique arc set which contains a given arc of the network if and only if that arc lies on a directed path from an origin node to one of the other nodes in the node set. In particular, it is easy to see that the arc set will consist precisely of every arc which has two nodes of the node set as its endpoints. (This allows a directed path from an origin node back to itself.) Moreover, it is possible to determine the node sets (e.g., by an appropriate specification of "origins") so that the associated arc sets are disjoint and, in fact, create a natural partition of all of the arcs of the network. (The node sets, on the other hand, are not disjoint.)

We can further, if desired, identify a decomposition of the network so that each node set has precisely one origin. However, this will not be essential to the task of specifying an efficient shortest path algorithm for the network. Our goal rather, will be to constructively identify a decomposition of the form indicated (if one is not given a priori) and then to prescribe a simple extension of a known shortest path calculation which exploits this decomposition in a computationally advantageous manner.

By the way of an overview, the strategy underlying the constructuve creation of the desired decompositi n may be loosely described as follows. Our construction proceeds by "dividing" each of the previously indicated node sets (other than the first and the last) into three

components x_{i-1}, A_i, x_i such that for i even the origin nodes of the sets all lie in A_i and there are no directed paths from nodes in x_{i-1} to nodes in A_i and no directed paths from nodes in x_i to nodes in A_i and the origin nodes of A_{i+1} lie in x_i. This refinement of the basic decomposition occurs in a completely natural manner and permits further improvement of the shortest path calculation for the total network. Moreover, the origin nodes are implicitly generated as a byproduct of the construction, and the method explicitly identifies sets of nodes which comprise minimal cut sets for cumulatively generated subnetworks. In particular, x_i is the minimal cut set for the subnetwork $A_1 \cup x_1 \cup A_2 \cup \cdots x_{i-1} \cup A_i$.

To begin, we construct the first node set $\bar{A}_1$ by selecting an arbitrary node to be its (unique) origin and then fanning out from successor to successor to find all nodes to which this node connects by a directed path. Assuming that $\bar{A}_1$ does not include all nodes of the network, there will remain some nodes that are not yet "reached" at this point. Since the network is connected some of the unreached nodes must have directed paths to some of the reached nodes. We next construct the sets A_1, x_1, and A_2 as follows. First find all nodes in $\bar{A}_1$ which have arcs incident into them from some node not in $\bar{A}_1$. Let x_1 denote this set of nodes in $\bar{A}_1$, and let $A_1 = \bar{A}_1 - x_1$. Next let A_2 initially consist of all nodes not in $\bar{A}_1$ which have arcs leading into x_1. Now add to A_2 all nodes that have arcs leading into nodes currently in A_2. The process continues in this fashion until no new nodes can be added to A_2. Thus A_2 consists of those nodes not in $\bar{A}_1$ which have directed paths leading into x_1. At this point either all nodes have been reached or the remaining unreached nodes have no directed path to the nodes in $\bar{A}_1 \cup A_2$. The set x_2 is then constructed to consist of the nodes not in $\bar{A}_1 \cup A_2$ which have arcs leading into them from nodes in A_2. We then let $\bar{A}_2 = x_1 \cup A_2 \cup x_2$.

Next $\bar{A}_3$ is constructed to consist of nodes that can be reached via a directed path from nodes in x_2. Using $\bar{A}_3$ we construct A_3, $\bar{A}_3$, and x_3 as follows. First find

all nodes in $\tilde{A}_3$ which have arcs incident into them from nodes not in $x_2 \cup \tilde{A}_3$. Let x_3 denote this set of nodes and define $A_3 = \tilde{A}_3 - x_3$ and $\bar{A}_3 = x_2 \cup A_3 \cup x_3$.

Continuing in this manner the nodes of the network N will be decomposed into m linearly overlapping sets $\bar{A}_1, \bar{A}_2, \ldots, \bar{A}_m$ such that $N = \bigcup_{i=1}^{m} \bar{A}_i$. At the same time we identify m - 1 node sets x_i where x_r is the minimum cut set for the subnetworks $A_1 \cup x_1 \cup A_2 \cup \ldots x_{r-1} \cup A_r$. Furthermore for odd i, there exists no directed paths between nodes in A_i and x_{i+1}. However, for i even, there may exist such paths. This decomposition is graphically depicted in Figure 2. In this type of decomposed network, if a directed path exists from some node k in $\bar{A}_i$, where i is an odd number, to another node j, then j is also in $\bar{A}_i$. Furthermore, if i is an even number, then j is either in $\bar{A}_{i-1}$, A_i, or $\bar{A}_{i+1}$. Thus if a network decomposes into several sets $\bar{A}_i$, considerable efficiency can be gained in obtaining all shortest paths. Networks which decompose into several sets have been termed by the authors regional networks.

While it is not obvious that networks will decompose into several $\bar{A}_i$ sets, the authors first observed that some randomly generated networks which were being used to test the computational efficiency of different minimum cost flow algorithms decomposed into regional networks. This observation was occasioned by the fact that the dual start method in [5] essentially performs this decomposition. Later the authors observed this fact in the water resource models discussed in the introduction and in product distribution models in the automobile and oil industries. In general, the likelihood that a network will lend itself to such a decomposition seems more transparent if one observes that in a large multi-source, multi-sink network not every sink can be reached (normally) by each source.

3 A Mathematical Procedure For Constructing the Decomposition

We shall now specify mathematically the procedure for decomposing a network N into sets of nodes $\bar{A}_i$, A_i, and x_i such that if i is odd then a node in $\bar{A}_i$ can at most reach other nodes in $\bar{A}_i$ and if i is even then a node in A_i can at most reach other

nodes in $\bar{A}_{i-1}$, A_i, or $\bar{A}_{i+1}$. And x_r is the minimum cut set for the subnetwork $A_1 \cup x_1 \cup A_2 \cup \ldots x_{r-1} \cup A_r$. In addition, when i is odd, there exists no directed paths between nodes in x_i and x_{i+1}. However, for i even, such paths may exist.

Using the following we define a decomposition procedure which accomplishes this segmentation of the network(or determines that the network is disconnected), we let

U = set of nodes in N which have not yet been assigned to some $\bar{A}_i$.

A = set of all admissible arcs in N.

n = number of nodes in N.

Initialization

Select an arbitrary starting node t. Initialize the sets V' = {t} and U = {i: i = 1, 2, ..., n, i $\neq$ t}. Set i = 1, $\tilde{A}_i$ = {t}, and proceed to I.1.

Part I

I.1 If the set V = {j: (h, j) $\in$ A, h $\in$ V', j $\in$ U} is empty, proceed to I.2; otherwise set U = U - V and V' = V.

I.2 If V is not empty, set $\tilde{A}_i = \tilde{A}_i \cup V$ and return to I.1. If U is empty, set $A_i = \tilde{A}_i$ and $\bar{A}_i = x_{i-1} \cup A_i$ (where $x_0 = \emptyset$) and stop. If V is empty, set x_i = {j: (k, j) $\in$ A, j $\in \tilde{A}_i$, k $\in$ U}, $A_i = \tilde{A}_i - x_i$ and $\bar{A}_i = x_{i-1} \cup A_i \cup x_i$ (where $x_0 = \emptyset$); if $x_i = \emptyset$ stop because the network is disconnected. Otherwise, set i = i + 1, V' = $\tilde{A}_i$ = {k: (k, j) $\in$ A, j $\in x_{i-1}$, k $\in$ U} and proceed to Part II.

Part II

II.1 If the set V = {h: (h, j) $\in$ A, h $\in$ U, j $\in$ V'} is empty, proceed to II.2; otherwise set U = U - V and V' = V.

II.2 If V is not empty, set $\tilde{A}_i = \tilde{A}_i \cup V$ and go back to II.1. If U is empty, set $A_i = \tilde{A}_i$ and $\bar{A}_i = A_i \cup x_{i-1}$ and stop. If V is empty, set x_i = {j: (k, j) $\in$ A, k $\in \tilde{A}_i$, j $\in$ U}, $A_i = \tilde{A}_i$ and $\bar{A}_i = x_{i-1} \cup A_i \cup x_i$; if $x_i = \emptyset$ stop because the network

is disconnected. Otherwise set $i = i + 1$, $V' = \tilde{A}_i = \{j: (k, j) \in A, k \in x_{i-1}, j \in U\}$ and proceed to Part I.

Since the decomposition procedure is a constructive process and is justified by straightforward considerations, we shall omit its proof for brevity.

4 A Shortest Path Calculation For the Decomposed Network

The problem of finding the shortest path between all pairs of nodes in a connected network which has been decomposed by the foregoing procedure can be solved quite simply by the following approach:

1. Using Hu's triple operation [7], determine the shortest distance between all pairs of nodes in each subset $\bar{A}_i$ for i odd. Using the distances determined in the preceding operation for nodes in the sets x_i, apply the triple operation to determine the shortest distance between all pairs of nodes in each subset $\bar{A}_i$ for i even.

2. Next determine the shortest distance from each node k in each even numbered set A_i to each node j in $A_{i-1} \cup x_{i-2}$ or $A_{i+1} \cup x_{i+1}$ using the following:

   a) if $k \in A_i$ and $j \in A_{i-1} \cup x_{i-2}$, set $d_{kj} = \min_{p \in x_{i-1}} \{d_{kp} + d_{pj}\}$

   b) if $k \in A_i$ and $j \in A_{i+1} \cup x_{i+1}$, set $d_{kj} = \min_{p \in x_i} \{d_{kp} + d_{pj}\}$

where d_{ij} denotes the shortest distance from node i to node j and by convention if no path exists from node i to node j, $d_{ij} = +\infty$.

The proof that the above procedure yields all shortest paths between any pair of nodes in the network follows directly from the decomposition procedure.

Recall that if i is odd, then any node $k \in \bar{A}_i$ can only reach other nodes in $\bar{A}_i$. Thus applying Hu's triple operation [7] to $\bar{A}_i$, yields all shortest distances from nodes in $\bar{A}_i$ to all nodes in N. If i is even, then any node $k \in \bar{A}_i$ can only reach nodes in $\bar{A}_{i-1}$, $\bar{A}_i$, and $\bar{A}_{i+1}$. Thus by using the shortest distances between nodes in the sets x_j and applying Hu's triple operation to $\bar{A}_i$ yields shortest distances

between every pair of nodes in $\bar{A}_i$. Further, Step 2a will yield the shortest distance between any node $k \in A_i$ and $j \in A_{i-1} \cup x_{i-2}$. This follows from the fact that d_{pj} and d_{kp} are already shortest path distances (since the nodes p and j are both in $\bar{A}_{i-1}$, and the nodes k and p are both in $\bar{A}_i$). Thus if the minimum is taken over all $p \in x_{i-1}$, then the shortest distance is obtained between k and j since any path from k to j must use a node in x_{i-1}. A similar argument may be used to verify that Step 2b will yield the shortest distance between any nodes $k \in A_i$ and $j \in A_{i+1} \cup x_{i+1}$. By the decomposition of the network, Steps 1, 2a, and 2b account for all possible directed shortest paths, and the justification is complete.

5 The Efficiency of the Shortest Path Algorithm

To examine the efficiency of the shortest path algorithm, we shall compare it to the modified algorithm developed by Yen. Assume that the node sets A, B, ..., H (m-sets) and A_1, A_2, ..., A_m each contain u nodes, and the subnetworks x_A, x_B, ..., x_G and x_1, x_2, ..., x_{m-1} each contain v nodes.

In Yen's algorithm, the total number of computations required is approximately

$$mu^3 + (m^2 + 6m - 7)u^2v + (2m^2 + 10m_{-20})uv^2 + (m^2 + 6m - 14)v^3 \qquad (1)$$

additions and comparisons. For the shortest path algorithm of this paper, Step 1 requires roughly $m(u + 2v)^3$ additions and comparisons and Step 2 requires roughly $m(u(u + v)v)$ additions and comparisons. The total number of computations is then less than

$$mu^3 + 7mu^2v + 13muv^2 + 8mv^3 \qquad (2)$$

additions and comparisons. Clearly the amount of computation required in (2) is significantly less than that required by (1). The number of comparisons required to decompose the network are omitted in the count of (2) since Hu and Torres and Yen assumed a priori the desired decomposition of their network. However, this number may be identified as follows. If one assumes that for each node there is a list containing

all its immediate successors and a list containing all its immediate predecessors, then the number of comparisons required to decompose the network is

$$2mu^2 + 5muv + 3mv^2 \qquad (3)$$

where these comparisons are simple logic comparisons (i.e., checking to see if something is equal to zero). It is clear that the amount of the computation of (2) and (3) combined is still significantly smaller than the amount of computation required in (1).

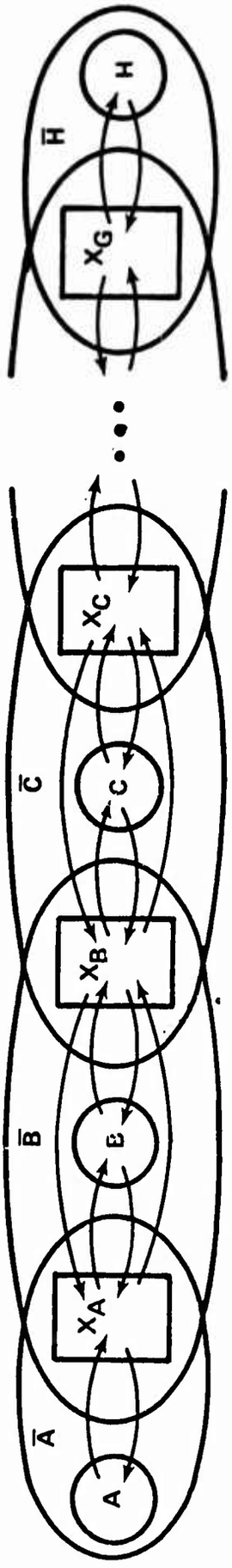
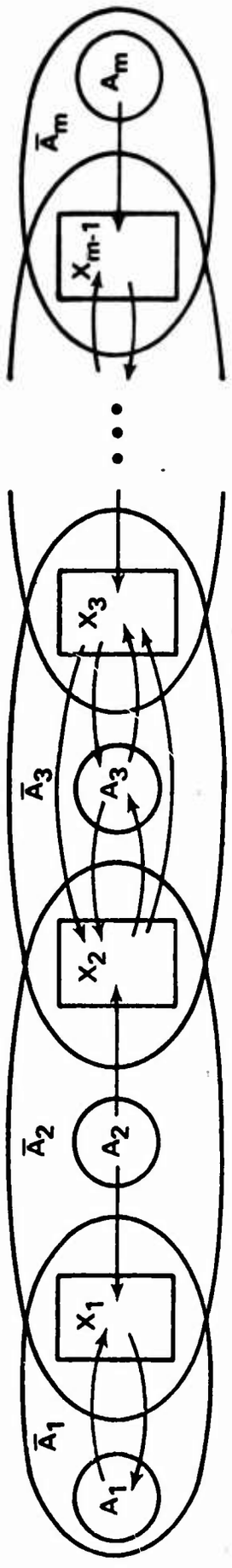


Figure 1


$\bar{A}_m$
A_m
X_{m-1}
$\bar{A}_3$
A_3
X_3
X_2
$\bar{A}_2$
A_2
X_1
$\bar{A}_1$
A_1

Footnotes

1. For an interesting survey of the types of shortest problems and their solution procedures see Dreyfus [2] and Elmaghraby [3].

2. Richard Barr, Fred Glover, and Darwin Klingman worked with the Texas Water Development Board to develop more efficient procedures for solving minimum cost flow networks and shortest path problems.

References

1. Dantzig, G. B., "All Shortest Routes in a Graph," Technical Report 66-3, Operations Research House, Stanford University, November 1966.

2. Dreyfus, S. E., "An Appraisal of Some Shortest-Path Algorithms," Operation Research 17, 395-412(1969).

3. Elmaghraby, S. E., "The Theory of Networks and Management Science, Part I" Management Science 17, 1-35 (September 1970).

4. Floyd, R. W., "Algorithm 97, Shortest Path," Communication of the ACM 5, 345 (1962).

5. Glover, Fred, D. Klingman, and A. Napier, "A One-Pass Algorithm To Determine A Dual Feasible Basic Solution For a Class of Capacitated Generalized Networks," Operation Research 20, 126-137 (January 1972).

6. Hu, T. C., "A Decomposition Algorithm for Shortest Paths in a Network," Operations Research 16, 91-102 (1968).

7. Hu, T. C., Integer Programming and Network Flows (Reading, Mass: Addison-Wesley Publishing Co. 1969).

8. Hu, T. C. and W. T. Torres, "Shortest in the Decomposition Algorithm for Shortest Paths in a Network," IBM Journal of Research and Development 13, 387-390 (1969).

9. Land, A. H. and S. W. Stairs, "The Extension of the Cascade Algorithm to Large Graphs," Management Science 14, 29-33 (1967).

10. Yen, J. Y, "On Hu's Decomposition Algorithm for Shortest Path in a Network," Operations Research 19, 983-985 (1971).